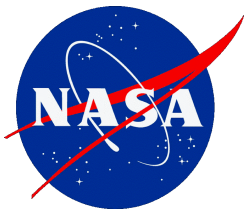
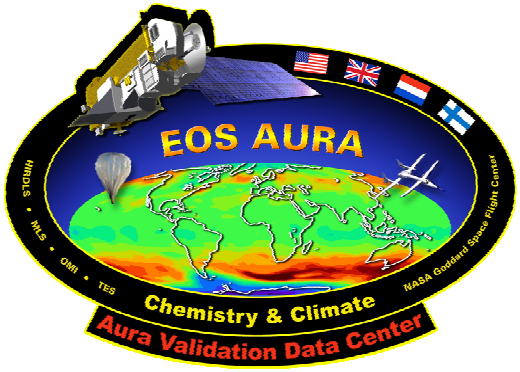


National Aeronautics and
Space Administration

Goddard Space Flight Center
Greenbelt, MD




# AVDC idlcr8hdf User's Guide

NASA NIWA-ERI Contract# NAS05903

Work Activity 916-011-1

Original: June 17, 2005
Revised: October 4, 2006
Version: 2.0

Document Profile Information (Inside Cover)

| Readme_idlcr8hdf.doc | |
|---|---|
| Title: | AVDC idlcr8hdf User's Guide |
| Version: | 2.0 |
| Type: | User manual |
| Audience: | Data providers |
| Author(s): | Ian Boyd (NIWA-ERI, i.boyd@niwa.com) and Bojan R. Bojkov (UMBC/GEST and NASA-GSFC, bojan.bojkov@gsfc.nasa.gov) |
| Status: | Final |
| Distribution: | Public |
| Location: | |
| | |
| Abstract: | This document describes how to use the idlcr8hdf.pro and idlcr8hdf.sav programs to create HDFv4.2 files suitable for submission to the AVDC, NDACC, and ESA ENVISAT (NILU) databases. |
| See Also: | N/A |

# Table Of Contents

1 Overview ..... 1

2 Running idlcr8hdf.pro on IDL LIC ..... 1

2.1 Run-time Options ..... 1
2.2 Command Line Input Parameters ..... 2
2.2.1 GA ..... 2
2.2.1.1 Session Memory Option ..... 2
2.2.1.2 Input File Option ..... 3
2.2.2 SDS ..... 3
2.2.2.1 Session Memory Option ..... 3
2.2.2.2 Input File Option ..... 4
2.2.3 TAV ..... 4
2.2.4 /H5 Option ..... 5
2.2.5 /AVK Option ..... 5
2.2.6 /Log Option ..... 5
2.2.7 /Popup Option ..... 5

3 Running idlcr8hdf.sav on IDL VM or IDL LIC ..... 6

4 The Metadata and Data ..... 7

4.1 Metadata ..... 7
4.2 Data ..... 9

5 The HDF Output File ..... 11

6 Procedures ..... 12

6.1 idlcr8hdf ..... 12
6.2 idlcr8hdf_Common ..... 12
6.3 Julian_Date ..... 13
6.4 JDF_2_Datetime ..... 13
6.5 Intro_Event ..... 13
6.6 Intro ..... 13
6.7 idlcr8hdf_Event ..... 13
6.8 Stop_With_Error ..... 13
6.9 Read_Tablefile ..... 13
6.10 Test_File_Input ..... 13

6.11 Read_Metadata ..... 14
6.12 Extract_And_Test ..... 14
6.13 Var_Units_Test ..... 14
6.14 Check_Metadata ..... 14
6.15 Set_Up_Structure ..... 14
6.16 Check_Min_Max_Fill ..... 14
6.17 Extract_Data ..... 14
6.18 Read_Data ..... 15
6.19 Find_HDF_Filename ..... 15
6.20 Make_An ..... 15
6.21 AVDC_HDF5_Write ..... 15
6.22 AVDC_HDF_Write ..... 15
7 Error Messages and Warnings ..... 15

7.1 idlcr8hdf Errors ..... 16
7.2 Read_Tablefile Errors ..... 17
7.3 Test_File_Input Errors ..... 17
7.4 Read_Metadata Errors ..... 17
7.5 Extract_And_Test Errors ..... 17
7.6 Var_Units_Test Errors ..... 18
7.7 Check_Metadata Errors ..... 18
7.8 Set_Up_Structure Errors ..... 18
7.9 Check_Min_Max_Fill Errors ..... 19
7.10 Extract_Data Errors ..... 20
7.11 Find_HDF_Filename Errors ..... 20
8 Acronyms ..... 20

9 Version History ..... 21

10 Testing ..... 23

11 References ..... 23

12 Contact Information ..... 24

# 1 Overview

NASA has developed the idlcr8hdf.pro and idlcr8hdf.sav programs for users of the Aura Validation Data Center (AVDC), Network for Detection of Atmospheric Composition Change (NDACC) and ESA ENVISAT (NILU) databases, to convert scientific measurements into HDFv4 files suitable for submission to the NDACC Data Handling Facility (DHF), the AVDC, and the ENVISAT data archives.

idlcr8hdf requires, as input, a Table Attribute Values (TAV) file, and either a Metadata file and Data file(s), or the Metadata Global Attributes in an array, and the Data and Metadata Variable Attributes in a heap structure using pointers. It writes the inputs to an HDF file as a multi-dimensional (maximum of 8 dimensions) Scientific Data Set. It uses predefined HDF routines for manipulating HDF files, and has been tested on IDL version 5.4 (which uses the NCSA HDF library v4.1r3) and IDL version 6.3[1]. Input can either be by command line (licensed version of IDL (IDL LIC)) or by DIALOG_BOX prompts (IDL LIC, or IDL Virtual Machine (IDL VM)).

For users who do not have an IDL license, IDL VM can be downloaded free from http://www.ittvis.com/idlvm. The resulting download is the full version of IDL. The user can install IDL VM only, or the full version (including IDL VM). The full version can only be used in 'Demo' mode without a license. IDL VM executes the binary version of the code (idlcr8hdf.sav). Refer to Chapter **3** for details.

# 2 Running idlcr8hdf.pro on IDL LIC

## *2.1 Run-time Options*

idlcr8hdf.pro can be run as a standalone program or called by another program, by compiling the program (.r idlcr8hdf) and calling the main procedure, idlcr8hdf, at the IDL prompt or from another program. The full command line, including parameters is:

**idlcr8hdf [,GA,SDS,TAV[,/H5][,/AVK][,/Log][,/Popup]]**

Possible run-time options are:

- Calling the procedure with no parameters (idlcr8hdf) – An 'Introduction' pop-up box will be displayed, and the user has the option of continuing with DIALOG_BOX prompts, or stopping the program. This is equivalent to running the program on IDL VM (see Chapter **3**).
- Session Memory Option – Inputs are a string array containing the Global Attributes, a heap structure containing the Variable Attributes and Data, and the TAV filename (if wanting to use a pop-up DIALOG_BOX to choose the TAV filename, then do not include a filename for 'TAV', e.g. idlcr8hdf, GA, SDS_Structure).

[1] To determine the HDF library used on your version of IDL call HDF_LIB_INFO, Version=ver.

- Input File Option – Inputs are Metadata, Data and TAV filenames. The data file input can be a string array containing the names of multiple data files, or a scalar string giving the name of a file spec (e.g. 'input*.data'), or a scalar string giving the name of a single file (e.g. idlcr8hdf, 'Metafile', ['Datafiles'], 'TAVfile' or idlcr8hdf,'','','' etc). If no input filenames are provided then pop-up DIALOG_BOXES will prompt for the three file inputs (using the IDL function DIALOG_PICKFILE), otherwise the program tests for the existence of the files, and if the files are not valid, opens DIALOG_BOXES to prompt for input.

If intending to use the DIALOG_PICKFILE function to select input files, the user can set the 'Path', 'File' and 'Filter' function parameters to reflect the directory and filename structure on his/her computer. This function is used in the idlcr8hdf procedure.

The output HDF file will be named based on the contents of the Metadata file, and placed in the program directory. If an HDF file of the same name already exists, it will be overwritten without any warnings. All integrity checks made by the program are done before the HDF file is opened for output. If an error is found then the program will immediately stop.

If input is in the form of an array and a heap structure, the user may want to free up memory, used by the SDS heap structure from previous calls to the program, by calling the PTR_FREE procedure in his/her own program (e.g. PTR_FREE, SDS.Data, SDS.VA). Input in this form can only contain information for one HDF file.

The program automatically checks for the type of input (session memory or files), by testing the SDS parameter. If it is a structure then it is assumed input is from session memory. If it is a scalar string or string array, then it is assumed that input is from files. If idlcr8hdf is called with no parameters, it is assumed that inputs will be in the form of files, if the user continues beyond the 'Introduction' dialog box.

If compiling idlcr8hdf on IDL6.1 or less, the module AVDC_HDF5_WRITE may show one or more compilation errors. This is to be expected as HDF5 write routines were introduced in IDL from version 6.2. The program will still run successfully, but the option to write HDF5 files will be unavailable. A program generated error message will be displayed if idlcr8hdf is called with the /H5 option from the command line. If the 'Introduction' display window is opened, the HDF5 Widget Button will be insensitive.

## *2.2 Command Line Input Parameters*

### 2.2.1 GA

#### 2.2.1.1 Session Memory Option

A one-dimensional string array of size NG_ATTS containing the Global Attributes (where NG_ATTS is the number of Global Attributes). Each entry to the array consists of one Global Attribute, defined through a metadata statement, in the form:

label = value

where label is the name of the attribute, and value is the corresponding value of that attribute. See Section **4.1** of this document, or the AVDC Metadata Guidelines document (download from http://avdc.gsfc.nasa.gov), for more information on the composition of the Global Attributes.

#### 2.2.1.2 Input File Option

This is a file containing the Metadata (both the Global and Variable Attributes). Each Metadata Attribute is defined through a Metadata statement in the file. A Metadata statement has the form:

label = value

where label is the name of the attribute, and value is the corresponding value of that attribute. See Section **4.1** of this document, or the AVDC Metadata Guidelines document (download from http://avdc.gsfc.nasa.gov), for more information on the Metadata file.

### 2.2.2 SDS

#### 2.2.2.1 Session Memory Option

This is a structure, using pointers, containing the user's scientific data. It contains N_SDS pointer arguments, where N_SDS is the number of datasets making up a single data measurement, and consists of the Data values and the Variable Attributes for each dataset. The heap structure sent to the procedure is expected to consist of at least two storage structures named XXX.Data and XXX.VA (where XXX is the heap structure name).

The datasets can either be in the form of formatted strings, or as the type given by the corresponding VAR_DATA_TYPE value for that dataset. The individual datasets can either have dimensions corresponding to the VAR_SIZE values, or a single dimension containing the total number of data values. For example, if the VAR_SIZE value for a dataset is 10;10, then the corresponding structure can either be represented as a two dimensional 10 x 10 array, or a single dimension of size 100.

The Variable Attributes for each dataset are represented as an NV_ATTS string array, where NV_ATTS is the number of attributes for the dataset. Each entry consists of a Variable Attribute, defined through a metadata statement, in the form:

label = value

where label is the name of the attribute, and value is the corresponding value of that attribute. See Section **4.1** of this document, or the AVDC Metadata Guidelines document (download from http://avdc.gsfc.nasa.gov), for more information on the composition of the Variable Attributes.

The heap structure must contain the Data and the Variable Attributes in the same order. For example, the Data in the structure *SDS(12).Data should correspond to the Variable Attribute information in the structure *SDS(12).VA.

More information on the connection between the data, as shown in the heap structure, and the indices of a multi-dimensional variable, is given in Section **4.2**.

#### 2.2.2.2 Input File Option

This can be a single filename containing the user's scientific data (e.g. measurements or model data), or a set of filenames described by a single file spec (e.g. input*.data), or a set of single filenames saved in a string array. The file(s) could be the output from an Excel spreadsheet, or from another program, which has converted the user's original data to an ASCII text format.

Each line has one value only. The value is either a header value or a data value. The header value is a text string containing the name of a variable corresponding to a VAR_NAME attribute value in the metadata file (without any quotes around it). See section **4.2** for more information on the composition of the data file.

### 2.2.3 TAV

The Table Attribute Values (TAV) file is an ASCII file containing tables of all legal values for a set of metadata attributes. The file is organized as a set of tables, each starting with the name of the table followed by a set of lines, each beginning with the equal character (=). The most recent open (non-encrypted) version of the TAV file can be downloaded from http://avdc.gsfc.nasa.gov.

This program identifies the version number of the TAV file and performs integrity (consistency) checks on the contents of the metadata file based on the values given in the file.

Non-comment lines in the file can be divided into two categories:

- A label line containing the name of a table
- A legal value line containing an equal character (=) followed by an allowed value of the corresponding metadata attribute (label)

An example of a label line and a set of corresponding legal value lines could be:

```
! This is a comment
VAR_DATA_TYPE
=REAL
=DOUBLE
=LONG
=INTEGER
# Another comment
```

If an attribute label is in the TAV file, only values listed in the legal values list are allowed. On the other hand, if an attribute label is not in the TAV file, any value is allowed. According to this definition the metadata statement, VAR_DATA_TYPE = REAL is allowed, but not, for example, VAR_DATA_TYPE = FLOAT.

The legal value line after the equal character may be either empty or may consist of only white space characters (blank or horizontal tab characters). In this case any empty string,

or a line consisting of only white space characters, will be a legal value for the corresponding attribute (label) in the metadata file.

Please refer to the AVDC web site (http://avdc.gsfc.nasa.gov) or the metadata guidelines document (Bojkov et al., 2002) for more information on the TAV file.

### 2.2.4 /H5 Option

Instead of creating a standard format AVDC HDF4 file, a compatible HDF5 file is created. This option is for the user's benefit only. It can be used, for example, together with idlcr8ascii to convert AVDC HDF4 files to HDF5 files. Global Attributes and Datasets (with associated Variable Attributes) are written to the 'root' group. VAR_DEPEND attribute information is included by means of an object reference pointer attached as an attribute to the dataset. This option is only available on IDL6.2 or greater.

### 2.2.5 /AVK Option

In the event that there are multi-dimensional Averaging Kernel data present in the measurement, the program will append a sentence to the relevant VAR_NOTES in the Metadata indicating the correct array order, if this option is chosen. The sentence reads: 'First three AVK values for the lowest altitude level are: x.xx x.xx x.xx'. If the dataset consists of more than one set of Averaging Kernels, then the sentence reads: 'First three AVK values for the lowest altitude level for the first measurement are: x.xx x.xx x.xx'. This option may be useful, for example, when using a metadata template file as input together with more than one data file. The program assumes that the first three values for the lowest altitude level would be equivalent to the first three values if the multi-dimensional array were written in a single column.

### 2.2.6 /Log Option

This option will cause the program to write input/output information to the file idlcr8hdf.log. If this file does not exist in the working directory then it will be created, otherwise the information will be appended to the file. The information is the same as that written to the IDL DE output log window and the pop-up display window (if this option is chosen), and also includes start and stop date and time of the program.

idlcr8hdf is set-up to display a run-time log in different ways, as follows:

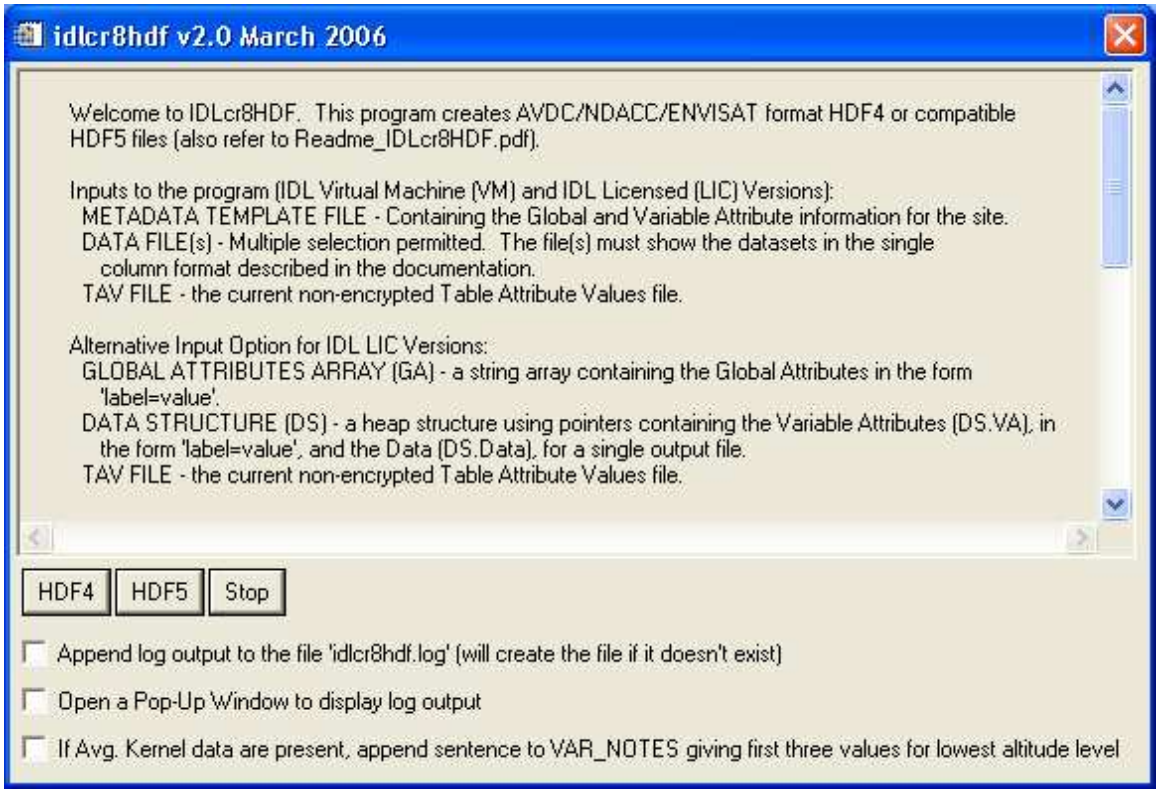
- If idlcr8hdf is called from another program, or from the IDL Command Input Line, then output will be to the IDL DE output log window.
- If idlcr8hdf.sav is opened in IDL VM, then the only way to see a log during run-time is to choose the Popup option.
- If the Log option is selected, output will also be to the file idlcr8hdf.log.
- If the Popup option is selected, output will also be displayed in a Pop-up window.

### 2.2.7 /Popup Option

This option will cause the program to write input/output information to a Pop-up window. Please note that, in some set-ups, choosing this option may stop you from being able to use other programs while idlcr8hdf is running. Select Finish to end the program.

# 3 Running idlcr8hdf.sav on IDL VM or IDL LIC

The file idlcr8hdf.sav is the binary version of idlcr8hdf.pro and can be run on IDL VM or IDL LIC. To run it on IDL LIC in the manner described in Chapter 2, the routines needs to be restored (restore,'idlcr8hdf.sav'), and the main procedure called (idlcr8hdf). To run on IDL VM, open IDL VM, and select idlcr8hdf.sav. The following 'Introduction' window will be displayed:



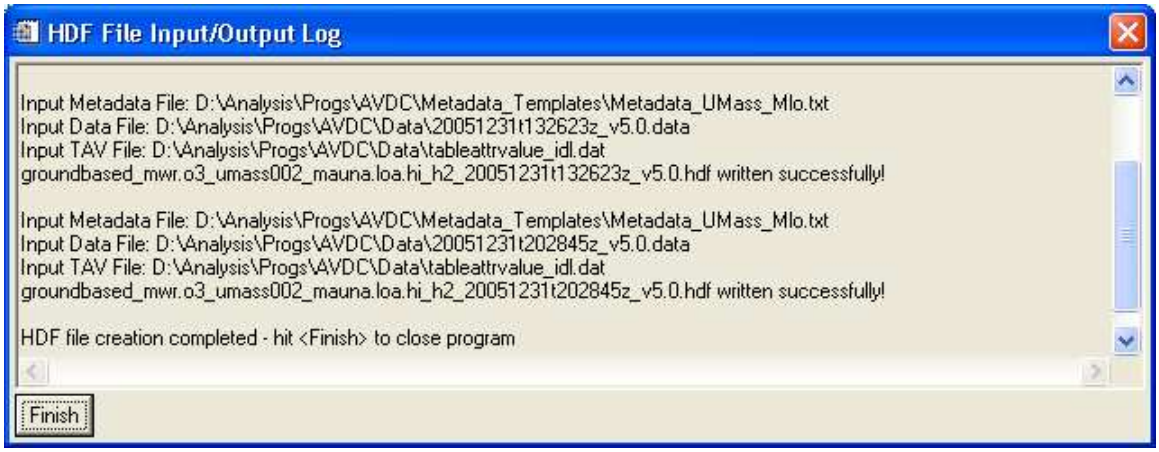

Output options are selected by a combination of exclusive and non-exclusive buttons as follows:

- HDF4 – output will be HDF4. This button is exclusive, meaning no other options are available once it is selected.
- HDF5 – output will be HDF5. This button is exclusive. It will be shaded if this option is not available. This is equivalent to choosing the /H5 option at the command line (refer to section 2.2.4).
- Stop or X – the Program will close.
- Log Output Option – This is equivalent to choosing the /Log option at the command line (refer to section 2.2.6). This option is non-exclusive, meaning other options are still available if this is selected.
- Popup Output Option – This is equivalent to choosing the /Popup option at the command line (refer to section 2.2.7). This option is non-exclusive.

- Avg. Kernel Option – This is equivalent to choosing the /AVK option at the command line (refer to section 2.2.5). This option is non-exclusive.

Once the HDF4 or HDF5 option is selected, the program will prompt for input file selection by Dialog Boxes. Inputs are: The Metadata Template file (refer to section 2.2.1.2 and 4.1), the Data file(s) (refer to section 2.2.2.2 and 4.2), and the TAV file (refer to section 2.2.3).

If all inputs are valid the program will perform integrity checks and, if successful, create the HDF file(s). If the Popup option is chosen, a log window, like the one below, shows progress through the operation. The program will stop once all files have been successfully created, or an error is detected. To close the program, select the Finish button.




# 4 The Metadata and Data

The metadata and data representation follows the AVDC/ENVISAT file reporting guidelines outlined by Bojkov et al., 2003.

## *4.1 Metadata*

The purpose of the metadata is to describe the users actual scientific data (e.g. measurements) using a set of descriptors or attributes. Each such attribute describes a certain part of the scientific data. Examples of attributes could be the name of the owner of the data, or the type of instrument that was used, or the measurement unit etc.

Each metadata attribute is defined through a metadata statement in the file. A metadata statement has the following form:

label = value

where label is a text string containing the name of the attribute, and value is a text string with the corresponding value of that attribute. An example of a metadata statement could be:

PI_NAME = Bojkov; Bojan R.

The semicolon character is used as a delimiter in the value text strings. Each item between the semicolons in a value is called a sub-value. The number of sub-values for a metadata attribute value is defined for each attribute (refer to the metadata guidelines for a detailed description of each attribute).

For many metadata attributes the concept of sub-values on the right hand side of the metadata statement is important since individual values will be checked for their legality against specified tables of allowed values. Tables of such legal values are defined in the TAV file. For a description of this file, see Section **2.2.3**.

Some metadata attributes describe date and time values. These are:

- DATA_START_DATE
- FILE_GENERATION_DATE

generally, and possibly:

- VAR_VALID_MIN
- VAR_VALID_MAX
- VIS_SCALE_MIN
- VIS_SCALE_MAX

The date and time value for these attributes can either be in the ISO8601 format (YYYYMMDDThhmmssZ) or in the MJD2000 (Modified Julian Date 2000) format on input to the program. The MJD2000 value is a double precision value counting the time (in fractional number of days) from 1 January 2000 at 00:00:00.0. Thus 20000101T000000Z is 0.0 MJD2000, while 20000101T120000 is 0.5 MJD2000. Date and time attribute values in the resulting HDF file are stored using MJD2000, apart from DATA_START_DATE and FILE_GENERATION_DATE, which are stored using ISO8601. If using ISO8601 format on input, the program will automatically convert the date and time values to MJD2000 where required (and vice versa). The HDF filename still uses the ISO8601 format for the part that describes the data start date.

There are five metadata attribute labels where the values can be determined directly by the idlcr8hdf code. It is therefore up to the user whether to include them in the input Metadata (although the attribute label and the '=' sign are still required).
These attribute labels are:

- DATA_START_DATE (determined from the data)
- FILE_NAME (determined from metadata values)
- FILE_GENERATION_DATE (determined by the program)
- FILE_META_VERSION (determined by the program)

- VAR_SI_CONVERSION (linked to the VAR_UNITS/UNIT_PREFIX values)

Most of the attribute values do not change from measurement to measurement, which makes it possible to use a single metadata template as input with multiple data files. Exceptions to this could be the DATETIME and related variables, e.g. XX_XX_START.TIME, XX_XX_STOP.TIME, where the VAR_VALID_MIN/MAX and VIS_SCALE_MIN/MAX values could change from measurement to measurement. To deal with this, the program looks for any variable name which has VAR_UNITS=MJD2000. If any of the four minimum or maximum metadata attribute values are missing, or the value is not numeric, then the program will determine these values based on the dataset entries which have VAR_UNITS=MJD2000. Please note that if numeric values for these attributes are present in the metadata file, they will not be replaced.

## *4.2 Data*

The data file is a plain ASCII file. As with the Metadata file, comment lines starting with an exclamation character (!) or a hash-mark character (#), and blank lines will be skipped over during input. Although an error will occur, and the program will stop, if a comment or blank line is inserted in the middle of a set of data values.

The data file is organized as follows:

```
<Header 1>
<Data value   1 for header 1>
<Data value   2 for header 1>
…
<Data value m1 for header 1>
<Header 2>
<Data value   1 for header 2>
<Data value   2 for header 2>
…
<Data value m2 for header 2>
…
<Header n>
<Data value    1 for header n>
<Data value    2 for header n>
…
<Data value mn for header n>
```

Each line has one value only. The value is either a header value or a data value. The header value is a text string containing the name of a variable corresponding to a VAR_NAME attribute value in the metadata file (without any quotes around it). For both the header line and on each of the data lines, any leading or trailing white-space characters will be stripped off the values. Thus, they do not need to start in position 1 of the lines.

A data file containing data for the variables:

- LONGITUDE

- LATITUDE
- ALTITUDE
- DATETIME
- TEMPERATURE

may look like this (example):

```
LONGITUDE
11.0
11.1
LATITUDE
60.0
61.0
62.0
ALTITUDE
9.6
12.4
18.9
23.5
DATETIME
0.0
1.0
2.0
3.0
4.0
TEMPERATURE
-2.7
-2.9
-3.4
-1.9
-0.7
…
-8.9
```

If TEMPERATURE has been made dependent upon the first four variables in the metadata file (i.e.):

```
VAR_DIMENSION = 4
VAR_SIZE = 2;3;4;5
VAR_DEPEND = LONGITUDE;LATITUDE;ALTITUDE;DATETIME
```

then it will contain a total of 2*3*4*5=120 values.

The connection between the data lines in the data and the indices of a multidimensional variable is always defined in IDL by letting the first index vary fastest, the second index next fastest, and so on.

To use the example above again, the connection between the data values after the header line for TEMPERATURE and the indices of the corresponding 4-dimensional variable TEMPERATURE is then the following:

```
…
1,1,1,1
2,1,1,1
1,2,1,1
2,2,1,1
1,3,1,1
2,3,1,1
1,1,2,1
2,1,2,1
…
2,3,4,1
1,1,1,2
2,1,1,2
…
2,3,4,5
```

HDF version 4 supports multidimensional variables with up to 8 dimensions.

The sequence of header values in the data file must be the same as the sequence of data variables as defined in the metadata file in the metadata attribute DATA_VARIABLES.

idlcr8hdf supports four different variable data types: INTEGER, LONG, REAL, DOUBLE. INTEGER is used for 16 bit integer data (integer*2). LONG is used for 32 bit integer data (integer*4). REAL is used for 32 bit single precision real data (real*4). DOUBLE is used for 64 bit double precision real data (real*8).

# 5 The HDF Output File

The program writes out an HDF file where all legal metadata attributes and their corresponding values are stored as HDF-attributes together with the SDS datasets from the data file. All attributes and their values are stored in the HDF file using text strings of variable length.

In addition to the user-defined attributes, the program also outputs to the HDF file a set of predefined HDF attributes, which NCSA (National Center for Supercomputing Applications) recommends should be part of any HDF file. These predefined attributes are shown in Table 1.

*Table 1: Predefined attributes and corresponding user-defined attributes.*

| Predefined attribute | Corresponding user-defined attribute |
|---|---|
| long_name | VIS_LABEL |
| units | VAR_UNITS |
| format | VIS_FORMAT |
| valid_range | VAR_VALID_MIN,VAR_VALID_MAX |
| _FillValue | VAR_FILL_VALUE |
| coordsys | VIS_SCALE_TYPE |

All attribute names, which start with either VAR (for variable), or VIS (for visualization), are defined as being dataset attributes (local attributes) belonging to one of the SDS's in the HDF file. All other attributes are defined in the HDF file as global attributes.

In addition to metadata attributes the HDF file contains the user's scientific data. There are four types of scientific data sets, which should almost always be present in the HDF file, namely:

- DATETIME
- LATITUDE or LATITUDE.INSTRUMENT
- LONGITUDE or LONGITUDE.INSTRUMENT
- ALTITUDE, ALTITUDE.INSTRUMENT, ALTITUDE.GPH, PRESSURE, DEPTH, DEPTH.INSTRUMENT

The DATETIME variable must always be declared as a 1-dimensional array with the data type DOUBLE (64-bits floating point number). The LATITUDE*, LONGITUDE* and any of the ALTITUDE*, DEPTH* or PRESSURE variables should always be declared as 1-dimensional arrays with the data type REAL (32-bits floating point number).

# 6 Procedures

The program is a standard IDL program consisting of 20 procedures and 2 functions. All the user data is read from external files or session memory.

A brief description of the procedures and functions is given below:

## *6.1 idlcr8hdf*

The main procedure called at run-time or by another external program. This procedure checks that the input is valid, calls the procedures that read the input and carry out the integrity checks, and finally calls the procedures that create the output HDF format file.

## *6.2 idlcr8hdf_Common*

This procedure defines the common blocks used by the program.

## 6.3 Julian_Date

A function which returns the Julian Day, or the date in MJD2000 format. Input can either be a numeric array in the form [YYYY,MM,DD[,hh[,mm[,ss]]]] or a string in the form YYYYMMDDThhmmssZ.

## 6.4 JDF_2_Datetime

A function which returns a floating point array in the form [YYYY,MM,DD,hh,mm,ss.sss], or a string in the form YYYYMMDDThhmmssZ, or a string in the form YYYYMMDDThhmmss.sssZ. Input can either be the Julian Day, or the date in MJD2000 format.

## 6.5 Intro_Event

Called by the Intro procedure. This procedure controls what happens when a Widget Button is selected in the Introduction display window.

## 6.6 Intro

Displays the Introduction display window, and allows the user to continue running the program after choosing from the available options.

## 6.7 idlcr8hdf_Event

Called by the Stop_With_Error procedure and the idlcr8hdf procedure. This procedure closes the Log display window and ends the program when the Finish Widget Button is selected.

## 6.8 Stop_With_Error

Called when an Integrity Check determines there is an error. This procedure displays the error in the log display window and the output log, and stops the program in a 'clean' manner (closes any open files etc).

## 6.9 Read_Tablefile

Identifies the version number of the TAV file. Also reads the FIELD/LABEL categories, the number of entries under each FIELD, and the FIELD entries into an NF x FCNT+2 string array called tab_arr (where NF is the number of FIELD categories, and FCNT is the maximum number of entries under the FIELDS). The additional two elements of the second dimension contain the name of the field and the number of values in the field.

## 6.10 Test_File_Input

Called by Read_Metadata. If a free text attribute is in the form of a filename, then checks that the file exists, and that the file size is less than or equal to the maximum permitted attribute length. Note: filename entries are not permissible under current AVDC Metadata Guidelines.

## 6.11 Read_Metadata

Puts the metadata contents into a string array called meta_arr. It ensures uniform input (e.g. making the ATTRIBUTE_NAME uppercase and deleting unwanted spaces etc), checks that the attribute value character length falls within allowable limits. It can also check for filenames for the Free Text Attributes (not permissible under current AVDC Metadata Guidelines).

## 6.12 Extract_And_Test

Called by Check_Metafile. Extracts the relevant portions of the attribute values from the metadata file and the TAV file and compares them.

## 6.13 Var_Units_Test

Called by Check_Metafile. Checks that the values given for VAR_UNITS are valid (equivalent to a base unit or a unit prefix and base unit) and independently calculates the VAR_SI_CONVERSION. If the VAR_SI_CONVERSION value is not present in the metadata, it automatically adds it, otherwise it checks that it matches the metadata value and, if not, it will replace it and write a warning message to the log.

## 6.14 Check_Metadata

Checks that metadata values for attribute names matching those in the TAV file are legal (i.e. included in the file). In some cases it will also make corrections to the metadata entry e.g. match VAR_SI_CONVERSION values to the VAR_UNITS value.

## 6.15 Set_Up_Structure

Performs additional checks on the metadata (for attributes not in the TAV file), and sets up a storage structure for the data based on the metadata values.

## 6.16 Check_Min_Max_Fill

Called by Read_Data. Does the following:

- Checks that the data values fall within VAR_VALID_MIN and VAR_VALID_MAX values (except for fill values).
- Checks that VAR_FILL_VALUE, VAR_VALID_MIN, VAR_VALID_MAX, VIS_SCALE_MIN, VIS_SCALE_MAX and data values fall within the range of the given data type.
- Checks that the VAR_FILL_VALUE falls outside the VAR_VALID_MIN or VAR_VALID_MAX values.
- Checks that VAR_FILL_VALUE, VAR_VALID_MIN, VAR_VALID_MAX, VIS_SCALE_MIN, VIS_SCALE_MAX and data values fall within the range described by the VIS_FORMAT value.
- Returns the data in an array of type determined by the VAR_DATA_TYPE attribute, and to the precision given in the VIS_FORMAT attribute.

## 6.17 Extract_Data

Called by Read_Data. This procedure extracts a specified dataset from the input data.

### 6.18 Read_Data

This procedure reads the input data, and after carrying out integrity checks, writes the datasets to a data structure for writing to the HDF file. This procedure also fills in missing VAR_VALID_MIN/MAX and VIS_SCALE_MIN/MAX values for datasets with VAR_UNITS=MJD2000 and, if requested, adds a sentence to VAR_NOTES giving array order for Averaging Kernel datasets.

### 6.19 Find_HDF_Filename

Does the following:

- Compares the DATA_START_DATE (if present) with the first entry under DATETIME in the data file. If necessary creates/changes the DATA_START_DATE to match the DATETIME entry, and puts it in ISO8601 format.
- Compares the filename created from the DATASET_ATTRIBUTES with that under the FILE_NAME entry (if present). If necessary creates/changes the FILE_NAME entry to match the created filename.
- If necessary creates/changes the FILE_GENERATION_DATE to ISO8601 format.

### 6.20 Make_An

Called by the two HDF write procedures in the event that a free text attribute entry is a filename. This procedure reads the contents of the file, and writes it to the HDF file as an SDS attribute. It can also be set up to write the file contents to the HDF file as a File or Data annotation (HDF_AN), although this feature is not currently implemented.

### 6.21 AVDC_HDF5_Write

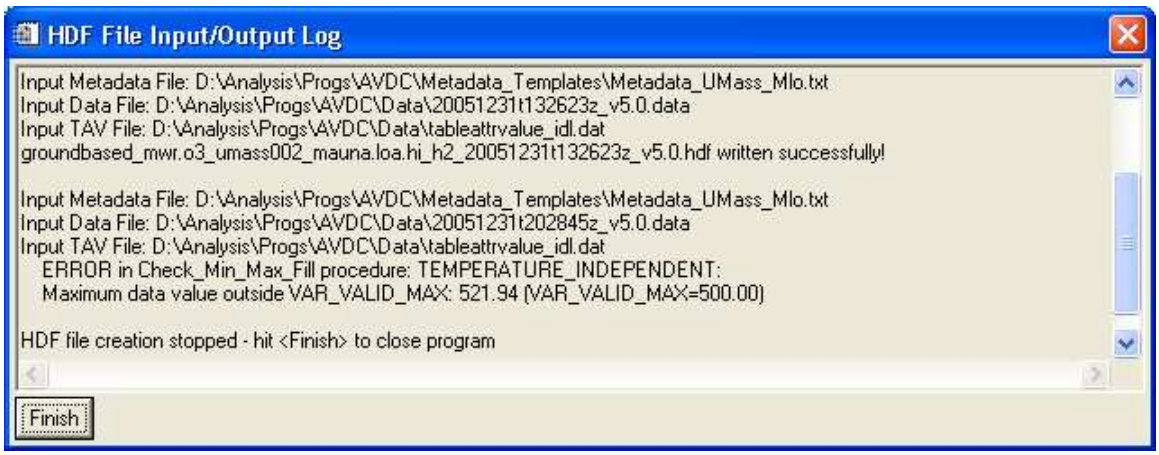
Called by AVDC_HDF_Write if the HDF5 option is chosen. Writes the Global and Variable Attributes and Data to an HDF5 file.

### 6.22 AVDC_HDF_Write

Writes the Global and Variable Attributes and Data as a Scientific Dataset to an HDF4 file according to AVDC guidelines, or calls AVDC_HDF5_Write if this option is chosen.

## 7 Error Messages and Warnings

Many of the procedures include integrity checks on the contents of the metadata and data. If an error is detected then the program will immediately stop, and an error message will be appended to the output log as well as the pop-up logging window or a separate pop-up box. The message includes the name of the procedure where the error was detected, a description of the error and the affected portion of the input file, for example:




In this case, TEMPERATURE_INDEPENDENT is a variable attribute. The procedure is checking the data values along with the VAR_VALID_MIN, VAR_VALID_MAX, VAR_FILL_VALUE, VIS_SCALE_MIN and VIS_SCALE_MAX metadata values, and has determined that a data value exceeds the VAR_VALID_MAX value given for this attribute.

In the case of warnings, the program continues to run, but a message appears in the log display window and output log. A warning message occurs when the program detects an error in the metadata file but is able to correct it.

All checks are carried out before the HDF file is created and opened. The checks are not complete, and do not guarantee that the resulting HDF file will be accepted by the database.

Below is a list of the possible errors and warnings, with further explanation. The list is sorted by procedure.

## *7.1 idlcr8hdf Errors*

- '**Metadata, Data and/or Table Attribute Values file selection not valid**'. If input is from files, the procedure checks for the existence of the three input files, and can also prompt the user for the names of the files. If any of the three files are not valid (e.g. are not found, or no input), then this message will be generated.
- '**No valid Table Attribute Values file selected**'. As above, but called when input from session memory is used.
- '**Global Attributes Array is not of type string, or is not one dimensional**'. If input is from session memory then checks that the Global Attribute array is of type string and has one dimension.
- '***SDS heap/SDS.Data/SDS.VA* structure is not valid**'. If input is from session memory then uses the N_Tags function to check for the existence of structure tag expressions, uses the Tag_Names function to check that tags names VA and DATA are present, and uses the PTR_Valid function to verify the validity of the pointer arguments to the heap structure.
- '**Number of Datasets does not match the number of Variable Attributes in the heap structure**'. If input is from session memory then checks that the

number of Datasets in the heap structure matches the number of Variable Attributes.

- '**IDLvX.X does not support HDF5 Write Routines**'. The IDL version must be 6.2 or greater if the /H5 option is chosen.

## 7.2 Read_Tablefile Errors

- '**Table Attribute Values file version not found with the search criteria used by this program**'. The program does a search on the text '! Version' to determine the file version of the TAV file. This is used as an entry under the FILE_META_VERSION global attribute.

## 7.3 Test_File_Input Errors

- '**Syntax of Filename entry incorrect**'. When listing a filename as a value for a Free Data Attribute, it needs to be in the form FILE("filename").
- '**Filename entry not found or not usable**'. The procedure checks that the file exists and can be read.
- '**File size is too large**'. The procedure checks that the file size is within the allowable limit given in the code (currently set to 4096 bytes).

## 7.4 Read_Metadata Errors

- '**Metadata line entry not valid ('=' not detected)**'. The procedure attempts to separate a line entry into its label and value components, which are separated by an '=' sign.
- '**Value contains too many characters**'. The procedure checks the number of characters in the free text entries, as well as the DATA_VARIABLES list, against a limit given in the code (currently 4096 characters).
- '**Does not match the expected Global Attribute Label**'. Tests against the list of Global Attribute Labels given in the procedure.
- '**Incorrect total number of Variable/Visualization Attributes**'. Checks that the total number of non-comment Metadata Attribute lines is as expected – based on the number of variables * the number of attributes per variable.
- '**Variable/Visualization attribute label is not valid**'. Compares the input attribute label against the list of Data Attribute Labels given in the procedure.
- '**No valid Metadata in command line parameter file or arrays**'. The procedure cannot identify any valid metadata information in command line inputs.
- '**This attribute label already read in for VAR_NAME=xxx**'. Checks to see if an attribute for a particular variable name has already been read in.

## 7.5 Extract_And_Test Errors

- '**Number of (sub-)values for this attribute should be xx**'. The number of values or sub-values being tested against the TAV file values is not correct.
- '**Doesn't match any values in Table Attribute Values file**'. The code is unable to find a match between a metadata value (or set of sub-values), and the list of possible values given in the TAV file.

## 7.6 Var_Units_Test Errors

- '**Doesn't match any values in Table Attribute Values file under VAR_UNITS/UNIT_PREFIX**'. The code is unable to find a match between the metadata value (or set of sub-values), and the list of possible values given in the TAV file.

## 7.7 Check_Metadata Errors

- '**No or Invalid value for this ATTRIBUTE**'. Relevant for metadata attribute values listed in the TAV file. This error will occur if no value exists or if there is more than one equal sign in the value.
- '**Number of (sub-)values for this attribute should be xx**'. The number of values or sub-values being tested against the TAV file values is not correct.
- '**AFFILIATION value doesn't match NAME value**'. If the ORIGINATOR field is present in the TAV file, the program checks that the AFFILIATION given in the metadata matches up to an AFFILIATION next to the NAME in the TAV file.
- '**WARNING: PI/DO/DS_ADDRESS/EMAIL values in Metadata replaced with those from the Table Attribute Values file**'. If values for these attributes are present in the metadata and the ORIGINATOR field is present in the TAV file, they are compared to the values in the TAV file derived when checking for the PI/DO/DS_NAME/AFFILIATION. If they are not the same, then the metadata values are replaced with those from the TAV file.
- '**WARNING: VAR_SI_CONVERSION values in Metadata replaced with those from the Table Attribute Values file**'. If the VAR_SI_CONVERSION sub-values in the metadata are present, they are compared to the values in the TAV file under VAR_UNITS (and possibly UNIT_PREFIX). If they are not the same, then the metadata values are replaced with those from the TAV file.

## 7.8 Set_Up_Structure Errors

- '**Does not match the equivalent VAR_NAME in DATA_VARIABLES**'. The procedure checks that the variable name corresponds exactly (and in the correct order), to the list given under DATA_VARIABLES.
- '**Number of dimensions too high to make HDF file (max. allowed 8)**'. IDL/HDF4 can only handle arrays up to eight dimensions.
- '**Number of attribute values does not match VAR_DIMENSION**'. Checks that the number of values in VAR_SIZE and VAR_DEPEND matches the dimension value.
- '**VAR_DEPEND value not 'CONSTANT/INDEPENDENT', and doesn't match a previous VAR_NAME**'. If the listed variable is dependent on another variable (e.g. DATETIME), then the VAR_DEPEND attributes must be listed before the dependent variable attributes.
- '**VAR_DEPEND variable name must have VAR_DEPEND value of 'CONSTANT/INDEPENDENT'**. The given variable name(s) must, in turn, have a VAR_DEPEND attribute value of CONSTANT or INDEPENDENT.

- '**VAR_DEPEND variable name must have matching VAR_SIZE value**'. The given variable name(s) must have the same number of data points (as indicated by VAR_SIZE), as the variable name being checked.
- '**Data type not INTEGER, LONG, REAL, or DOUBLE**'. The VAR_DATA_TYPE cannot be identified as one of the four allowable types.
- '**Unexpected number of ATTRIBUTE values extracted**'. Checks that there is the correct number of values needed to obtain the array size information.
- '**VAR_DATA_TYPE should be DOUBLE for VAR_UNITS=MJD2000**'. The VAR_DATA_TYPE value is not equal to DOUBLE.
- '**VAR_UNITS should be MJD2000 for VAR_NAME=DATETIME**'. The VAR_UNITS value is not equal to MJD2000.
- '**VAR_FILL_VALUE not a valid value**'. The program checks that the VAR_FILL_VALUE for attributes which have VAR_UNITS=MJD2000, is a valid number if it is in ISO8601 format.
- '**Type conversion error. Entry is not a valid number**'. The program cannot convert the attribute value from a string to its numeric value.

## 7.9 Check_Min_Max_Fill Errors

- '***Data Value* not valid, or does not match VAR_DATA_TYPE**'. Where ***Data Value*** could be VAR_VALID_MIN, VAR_VALID_MAX, VAR_FILL_VALUE, VIS_SCALE_MIN, VIS_SCALE_MAX or a Data Value. The format (label=value) of the metadata file entry is not correct, or the format does not match the data type (e.g. VAR_DATA_TYPE=INTEGER or LONG requires VIS_FORMAT=I*w*, where *w* is a number).
- '***Minimum/Maximum* data value outside VAR_VALID_*MIN/MAX***'. A data value falls outside the range given by VAR_VALID_MIN or VAR_VALID_MAX.
- '**Fill data value not outside VAR_VALID_MIN or VAR_VALID_MAX**'. The VAR_FILL_VALUE has to be outside the valid minimum or maximum values.
- '***Data Value* outside the data type range**'. Where ***Data Value*** could be VAR_VALID_MIN, VAR_VALID_MAX, VAR_FILL_VALUE, VIS_SCALE_MIN, VIS_SCALE_MAX or a Data Value. For example, if a data value is 40000, but VAR_DATA_TYPE=INTEGER (maximum allowable value of 32767).
- '***Data Value* outside the range given by VIS_FORMAT**'. Where ***Data Value*** could be VAR_VALID_MIN, VAR_VALID_MAX, VAR_FILL_VALUE, VIS_SCALE_MIN, VIS_SCALE_MAX or a Data Value. The data value cannot be generated according to the given format.
- '**Difference between D/E/F[w.d] values too small**'. By IDL/Fortran definition, the w and d values must be sufficiently different, depending on the format and, in the case of 'E', the operating system. This difference is also dependent on if there are negative data values present.
- '**DATETIME value cannot be a VAR_FILL_VALUE**'. Because DATETIME is an Independent variable it should not contain Fill Values.
- '**DATETIME values not in chronological order**'. If more than one DATETIME value is present, the program checks that the dataset is in chronological order.

- '**Type conversion error. Entry is not a valid value/format code**'. Cannot convert a data value or the numeric part of the VIS_FORMAT to a valid number.

### 7.10 Extract_Data Errors

- '**EOF encountered but not expected**'. If input is from files then this could occur when searching for the correct variable name header, or when reading in the data and encountering the EOF when expecting more data values.
- '**EOF expected but not encountered**'. If input is from files then extra non-comment lines were found in the file after reading in all the expected data.
- '**ISO8601 format not valid**'. Occurs when trying to convert a Datetime value, which the program thinks is in ISO8601 format, to MJD2000.
- '**Number of data values does not match product of VAR_SIZE**'. The number of values in the data structure or file for the given dataset does not match the total expected number from the VAR_SIZE attribute.

### 7.11 Find_HDF_Filename Errors

- '**Incorrect number of sub-values under DATA_nnn**'. Checks that the number of sub-values, corresponding to the attributes DATA_DISCIPLINE/_SOURCE/ _LOCATION/_LEVEL/_FILE_VERSION, is correct.
- '**ISO8601 format not valid**'. Occurs when trying to convert a Datetime value, which the program thinks is in ISO8601 format, to MJD2000.
- '**Type conversion error. MJD2000 entry is not valid**'. Occurs when trying to convert a Datetime value, which the program thinks is in MJD2000 format, from a string to a double precision value.
- '**WARNING: Date in DATA_START_DATE does not match first DATETIME entry**'. The program compares the first DATETIME, from the data, to the DATA_START_DATE (if present). If they are not the same then the DATA_START_DATE is replaced with the DATETIME value.
- '**WARNING: Filename determined from DATASET ATTRIBUTES does not match FILE_NAME entry under FILE ATTRIBUTES**'. The program independently determines the filename from the DATASET ATTRIBUTES. If this is different from the FILE_NAME value (if present), then the filename determined by the program is used.

## 8 Acronyms

| | |
|---|---|
| AVDC | Aura Validation Data Center |
| DHF | The NDACC's Data Handling Facility |
| ENVISAT | European Space Agency satellite |
| ESA | European Space Agency |
| HDF | Hierarchical Data Format |
| IDL | Research Systems, Inc., Interactive Data Language |
| ISO | International Organization for Standardization |

| | |
|---|---|
| MJD2000 | Modified Julian Date 2000 |
| NCSA | National Center for Supercomputing Applications |
| NDACC | Network for the Detection of Atmospheric Composition Change (ex-NDSC) |
| NILU | Norwegian Institute of Air Research |
| RSI | Research Systems, Inc. |
| SDS | Scientific Data Set data object in HDF |
| TAV | Table Attribute Values file used to create HDF |

# 9 Version History

v1.0 – Initial version release June, 2005.

v1.1 – Bug fixes and improvements as follows (release August, 2005):

- Change procedure heading format and order to allow library calls.
- Make VAR_SI_CONVERSION values match the VAR_DATA_TYPE, and add fix for VAR_SI_CONVERSION calculation rules e.g. if VAR_UNITS=cm m-3 then VAR_SI_CONVERSION=0;0.01;m-2.
- Additional checks of VAR_DEPEND values to reduce possibility of errors when creating the HDF file.
- Check that VAR_UNITS=MJD2000 for VAR_NAME=DATETIME.
- Improve checks and error messages when reading in data from a file.
- Add fix for converting a scalar data structure value to an array.
- Check that the first DATETIME value is the lowest when VAR_SIZE is greater than 1.
- Fix method that the program uses to check for integer overflow e.g. where VAR_DATA_TYPE=INTEGER but a data value is greater than 32767.
- Add fix for the situation where only fill values are present in a dataset.
- Change the format that DATA_START_DATE and FILE_GENERATION_DATE values are saved in the HDF file from MJD2000 to ISO8601.
- Replace JULDAY function calls (i.e. JULDAY(1,1,2000,0,0,0)) with the actual value (2451544.5D).
- Fix to write the DATA_FILE_VERSION value to the filename in its entirety.
- Add fix to avoid rounding errors when checking and writing double precision values.

v1.11 – Bug fixes and improvements as follows (release November, 2005):

- Change order of the routines to avoid problems during compilation.
- Write VAR_VALID_MIN, VAR_VALID_MAX, VAR_FILL_VALUE, VIS_SCALE_MIN, VIS_SCALE_MAX and Data Values to file using the precision and format given by VIS_FORMAT.
- Fix problems with detection of data values outside VAR_VALID_MIN or VAR_VALID_MAX, or outside the range given by VIS_FORMAT, and improve corresponding error messages.

v2.0 – Modifications to the code as follows (release October, 2006):

- Introduce option to write HDF5 versions of the AVDC HDF4 files (needs IDL6.2 or greater).

- Make the code suitable for running on a licensed version of IDL (using the .pro and .sav versions of the code) and on IDL Virtual Machine (using the .sav version of the code).
- Change the command line keyword options. Remove the /Help (window now opened if there are no command line parameters) and /File options (now automatically identifies whether input is from session memory or files), and add the /H5 (write HDF5), /AVK (for Averaging Kernel Datasets add sentence to VAR_NOTES indicating averaging kernel order), /Log (append input/output information to a log file), and /Popup options (append input/output, error and warning information to a pop-up display window).
- If input is from files, the program can now handle multiple data files (either as a string array or as a file spec), together with a Metadata template file and the TAV file.
- Input/output information as well as errors and warnings included in the IDL DE output log window.
- Program can fill in missing VAR_VALID_MIN/MAX and VIS_SCALE_MIN/MAX attribute values for datasets where VAR_UNITS=MJD2000.
- Add checks to DATETIME dataset – that the entries are in chronological order, if the dataset contains more than one value, and that the dataset does not contain Fill Values.
- Help window becomes an Introduction window, and the user has the option of continuing to run the program with file inputs.
- Input data now written to a structure instead of arrays, after the integrity checks have been performed and the datasets have been converted to the given data type and precision.
- The HDF4 file now not closed between writing the global attributes and the variable attributes to the file. This may result in a smaller file size.
- Fixes a problem with the calculation of VAR_SI_CONVERSION, when there is more than one VAR_UNITS sub-value with a unit prefix. The portion of the code that performs these checks also made clearer.
- Permits additional Global Attributes. If the attribute label DATA_TYPE is detected it is automatically changed to DATA_LEVEL.
- Permits the additional Variable Attribute VAR_MONOTONE.
- Input Variable and Visualization attributes can be in any order – they will be put in a standard order when writing to the HDF file.
- Fix to ensure that VAR_NAME, DATA_VARIABLES, and VAR_DEPEND values are case sensitive.
- Fixes a problem, in a multi-dimension array, when the last dimension is 1, e.g. VAR_SIZE=41;1 (previously the array size would be automatically reduced by IDL).

# 10 Testing

This program was written and tested on a Windows platform using RSI IDL version 6.3, and NCSA HDF library v4.1r5. It has also been tested on Linux (Debian GNU/Linux 2.6) and Apple Macintosh OSX (10.3, 10.4) platforms using RSI IDL version 6.1.

# 11 References

B.R. Bojkov, De Mazière, M. and R. Koopman, Generic metadata guidelines on atmospheric and oceanographic datasets for the ENVISAT Calibration and Validation Project, Version 01R001, April 23, 2002.

ISO, "Representation of Dates and Times", ISO 8601:1988, International Organization for Standardization (ISO), Geneva, Switzerland, (1988).

Research Systems, Inc, "IDL 6.1 Quick Reference", RSI, July 2004, Boulder, CO 80301.

# 12 Contact Information

For comments and to report bugs, please contact:

Ian Boyd, NIWA Environmental Research Institute
Department of Astronomy
619 Lederle Graduate Research Centre, University of Massachusetts
710 North Pleasant St.
Amherst, MA 01003, USA
E-mail:i.boyd@niwa.com
Phone: (+01) 413-545-2713
Fax: (+01) 413-545-4223

and

Bojan R. Bojkov (AVDC Project Manager)
NASA Goddard Space Flight Center, Code 613.3
Greenbelt, MD 20771, USA
E-mail:Bojan.Bojkov@gsfc.nasa.gov
Phone: (+01) 301-614-6846
Fax: (+01) 301-614-5903